

08/10/00
JC907 U.S. PTO

8-164-00

A

Please type a plus sign (+) inside this box → ☐

PTO/SB/05 (4/98)
Approved for use through 09/30/2000 OMB 0651-0032
Patent and Trademark Office, U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

UTILITY PATENT APPLICATION TRANSMITTAL

(Only for new nonprovisional applications under 37 C.F.R. § 1.53(b))

Attorney Docket No. **RCC-001**
First Inventor or Application Identifier **Sujith Arramreddy**
Title **"Peripheral Component Interconnect Arbiter..."**
Express Mail Label No. **EL597198174US**

APPLICATION ELEMENTS

See MPEP chapter 600 concerning utility patent application contents.

ADDRESS TO: Assistant Commissioner for Patents
Box Patent Application
Washington, DC 20231

- ☐ * Fee Transmittal Form (e.g., PTO/SB/17)
(Submit an original and a duplicate for fee processing)
- ☒ Specification [Total Pages **16**]
(preferred arrangement set forth below)
 - Descriptive title of the Invention
 - Cross References to Related Applications
 - Statement Regarding Fed sponsored R & D
 - Reference to Microfiche Appendix
 - Background of the Invention
 - Brief Summary of the Invention
 - Brief Description of the Drawings (if filed)
 - Detailed Description
 - Claim(s)
 - Abstract of the Disclosure
- ☒ Drawing(s) (35 U.S.C. 113) [Total Sheets **5**]
- Oath or Declaration [Total Pages - ☐ Newly executed (original or copy)
- ☐ Copy from a prior application (37 C.F.R. § 1.63(d))
(for continuation/divisional with Box 16 completed)
 - ☐ **DELETION OF INVENTOR(S)**
Signed statement attached deleting inventor(s) named in the prior application, see 37 C.F.R. §§ 1.63(d)(2) and 1.33(b).

- ☐ Microfiche Computer Program (Appendix)
- Nucleotide and/or Amino Acid Sequence Submission (if applicable, all necessary)
 - ☐ Computer Readable Copy
 - ☐ Paper Copy (identical to computer copy)
 - ☐ Statement verifying identity of above copies

ACCOMPANYING APPLICATION PARTS

- ☐ Assignment Papers (cover sheet & document(s))
- ☐ 37 C.F.R. § 3.73(b) Statement ☐ Power of Attorney
(when there is an assignee)
- ☐ English Translation Document (if applicable)
- ☒ Information Disclosure Statement (IDS)/PTO-1449 ☒ Copies of IDS Citations
- ☐ Preliminary Amendment
- ☒ Return Receipt Postcard (MPEP 503)
(Should be specifically itemized)
- ☐ * Small Entity Statement(s) ☐ Statement filed in prior application
(PTO/SB/09-12) Status still proper and desired
- ☐ Certified Copy of Priority Document(s)
(if foreign priority is claimed)
- ☐ Other: _____

NOTE FOR ITEMS 1 & 13 IN ORDER TO BE ENTITLED TO PAY SMALL ENTITY FEES, A SMALL ENTITY STATEMENT IS REQUIRED (37 C.F.R. § 1.27), EXCEPT IF ONE FILED IN A PRIOR APPLICATION IS RELIED UPON (37 C.F.R. § 1.28).

16. If a CONTINUING APPLICATION, check appropriate box, and supply the requisite information below and in a preliminary amendment:

☐ Continuation ☐ Divisional ☐ Continuation-in-part (CIP) of prior application No: _____
Prior application information: Examiner _____ Group / Art Unit: _____

For CONTINUATION or DIVISIONAL APPS only: The entire disclosure of the prior application, from which an oath or declaration is supplied under Box 4b, is considered a part of the disclosure of the accompanying continuation or divisional application and is hereby incorporated by reference. The incorporation can only be relied upon when a portion has been inadvertently omitted from the submitted application parts.

17. CORRESPONDENCE ADDRESS

☒ Customer Number or Bar Code Label **022888** or ☐ Correspondence address below
(Insert Customer No. or Attach bar code label here)

Name	BEVER, HOFFMAN & HARMS, LLP				
Address	2099 Gateway Place, Suite 320				
City	San Jose	State	CA	Zip Code	95110-1017
Country		Telephone	(408) 451-5907	Fax	(408) 451-5908

Name (Print/Type)	Jeanette S. Harms	Registration No. (Attorney/Agent)	35,537
Signature		Date	8-10-00

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO Assistant Commissioner for Patents, Box Patent Application, Washington, DC 20231

1 PERIPHERAL COMPONENT INTERCONNECT ARBITER IMPLEMENTATION
2 WITH DYNAMIC PRIORITY SCHEME

3 Sujith Arramreddy

4 Ragu Appanagari
5

6 BACKGROUND OF THE INVENTION

7 Field of the Invention

8 The present invention relates to a peripheral
9 component interconnect (PCI) arbiter, and in particular to
10 a PCI arbiter with a dynamic priority scheme.
11

12 Discussion of the Related Art

13 A peripheral component interconnect (PCI) bus is an
14 industry standardized expansion bus that conveys much of
15 the information and signals of a computer system.
16 Optimally, when the computer system executes its
17 programming, information should flow as fast as possible to
18 ensure the computer is responsive to the user. To prevent
19 mistakes in the transmission of that information, a PCI bus
20 design includes a special logic circuit and associated
21 signals to control the flow of that information.

22 Specifically, a typical PCI bus allows a bus
23 controller, also called an arbiter, to control bus
24 transfers. A device that takes control of the bus to
25 handle its own transfer is termed a "master", whereas a
26 device that receives data from the master is termed a
27 "target". The arbiter uses an algorithm to determine which
28 master can take control of the bus and the time period of
29 that control.

30 Arbitration must resolve the competing goals of
31 fairness and priority. Fairness requires that one master

1 should not be allowed to monopolize the bus. However,
2 priority requires that, under certain circumstances,
3 predetermined masters should use the bus more often to
4 accomplish time critical goals. Some typical algorithms
5 used by arbiters are the Single-Level Round Robin, the
6 Multi-Level Round Robin, the Least Recently Used, and the
7 Priority Based approaches.

8 In the Single Level Round Robin approach, a small unit
9 of time, i.e. a quantum, is defined. All processes
10 (associated with specific masters) are put in a circular
11 queue. The arbiter follows the queue, and allocates the
12 master's use of the bus to accomplish the process for a
13 time interval of one quantum. Any new process is added
14 after the last process in the queue.

15 If the process finishes before the end of the quantum,
16 the master releases the bus voluntarily. However, if the
17 process is still running at the end of the quantum, the
18 master is preempted and the process is added to the end of
19 the queue. In either case, the arbiter assigns the bus to
20 the next process in the queue.

21 In the Multi-Level Round Robin approach, at least two
22 circular queues are formed. For example, assuming first
23 and second queues are formed, processes that use the bus
24 frequently are placed in the first queue and processes that
25 use the bus less frequently are placed in the second queue.
26 Processes in the second queue have equal access to the bus,
27 if in the same queue. However, the processes in the second
28 queue, as a group, have equal access to the bus as each
29 process in the first queue. In other words, the processes
30 of the second queue effectively form a "super process",
31 wherein the super process is deemed to be one of the
32 processes in the first queue. Thus, for every round of the

1 first queue processes, one process of the second queue is
2 performed. In this approach, if the process finishes
3 before the end of the quantum, the master releases the bus
4 voluntarily. However, if the process is still running at
5 the end of the quantum, the master is preempted and the
6 process is added to the end of the appropriate queue.

7 In the Least Recently Used approach, an arbitrary
8 queue is formed. The arbiter initially follows the queue
9 and allows each process to finish before allowing the next
10 master in the queue to get control of the bus. However, if
11 the arbiter receives a request for bus control from a
12 master not next in the queue, the arbiter gives control of
13 the bus (after the completion of the process running) to
14 the master that has least recently used the bus.

15 Finally, in a priority-based approach, the arbiter
16 determines bus control based solely on the priority of the
17 associated process performed by the master. In this
18 approach, each process completes before the next process is
19 initiated.

20 Each of the above-described approaches has its
21 disadvantages. For example, in both the Single- and Multi-
22 Level Round Robin approaches, a quantum may not allow a
23 master time to finish a critical process. Therefore,
24 completion of that critical process may take several
25 complete cycles of the queue, thereby introducing
26 significant inefficiency in the system.

27 In the Least Recently Used approach, processes that
28 are non-critical get more bus control than in other
29 approaches. Although this allows less frequently used
30 processes an opportunity to complete, it also necessitates
31 losing time resources for other more critical processes.

1 Therefore, this approach also frequently results in
2 significant inefficiencies in the system.

3 In the Priority Based approach, depending on the task
4 to be completed by the system, non-critical processes may
5 only rarely be run. Although these non-critical processes
6 may relate only to processes such as expansion bus
7 requests, sporadic or even non-completion of these
8 processes may contribute to some system inefficiency.

9 All of these approaches use static algorithms to
10 determine control of the bus. As noted above, each of
11 these static algorithms fails to provide the flexibility to
12 optimize system efficiency. Therefore, a need arises for a
13 flexible, arbitration scheme that optimizes system
14 efficiency.

16 SUMMARY OF THE INVENTION

17 The present invention provides a dynamic priority
18 scheme that uses information including the status of the
19 target and data availability in deciding which master
20 should be assigned ownership of a PCI bus, thereby
21 optimizing performance and utilization of the PCI bus.
22 Specifically, the present invention provides multiple
23 levels of master priority. In one embodiment, three levels
24 of priority are provided: HIGH, MEDIUM, and LOW.

25 Once a request from a master is posted, an arbiter in
26 the system issues a provisional grant signal to the master.
27 At this point, the arbiter in the system assigns the
28 requesting master a MEDIUM priority and forwards the
29 request to the target. The arbiter then determines if data
30 is available from the target. If data is available, then
31 the arbiter reassigns the requesting master a HIGH
32 priority. However, if data is not available, then the

1 arbiter reassigns the requesting master a LOW priority and
2 ignores the requesting master until the arbiter is notified
3 that data is available from the target.

4 In accordance with the present invention, each target
5 includes a memory interface to facilitate the
6 prioritization process. Specifically, upon receipt of a
7 memory access request from a master (via the arbiter), the
8 target stores this request in a request queue, which forms
9 part of the memory interface, and then proceeds to capture
10 the information needed to complete the access of the
11 memory. After the data is copied in the request queue
12 (i.e., the data is now available), the target generates a
13 master ID for triggering a status change of the requesting
14 master. In a preferred embodiment, the target generates
15 the master ID using the request from the master (called a
16 modified request). This master ID is then provided to the
17 arbiter.

18 After the arbiter receives the modified request, the
19 arbiter changes the priority of the master to HIGH and,
20 assuming the PCI bus is available and no other masters have
21 an earlier high priority, sends the requesting master a
22 final grant signal, thereby allowing the master to take
23 control of the PCI bus. Note that if the PCI bus is
24 currently being controlled by another master or if other
25 masters have an earlier high priority, then the arbiter
26 sends the final grant signal at the next earliest time
27 period after the process performed by the last controlling
28 master is complete.

29 To further increase the efficiency of the present
30 invention, the request queue may include an input/output
31 cache. A cache controller keeps the cache filled with data
32 or instructions that one or more masters are most likely to

1 need next. In this manner, information can be retrieved
2 without delay.

3 4 BRIEF DESCRIPTION OF THE DRAWINGS

5 Figure 1 illustrates a simplified diagram of a PCI bus
6 architecture.

7 Figure 2 illustrates a simplified diagram of a
8 request/grant architecture.

9 Figure 3 illustrates a flow chart of the priority
10 scheme for a requesting master.

11 Figure 4 illustrates an example prioritization process
12 including four requesting masters.

13 Figure 5 illustrates an exemplary request queue in
14 accordance with the present invention.

15 16 DETAILED DESCRIPTION OF THE DRAWINGS

17 Figure 1 illustrates a PCI bus architecture 100
18 including a PCI bus 106. Connected to PCI bus 106 are
19 masters 101A, 101B, and 101C. A microprocessor 105
20 delegates arbitration of PCI bus 106 to an arbiter 103
21 (coupled to microprocessor 105 via a local bus 108) and is
22 viewed by the system as another master in architecture 100.
23 At this point, unless microprocessor 105 reassumes control,
24 arbiter 103 assigns ownership of PCI bus 106 to requesting
25 PCI masters.

26 Targets 102A and 102B, which can be accessed by any
27 master, are connected to arbiter 103 via memory buses 107A
28 and 107B, respectively. For purposes of this description,
29 targets 102A and 102B include associated memories 104A and
30 104B, respectively.

31 In a PCI design, bus arbitration includes the
32 following steps. A master asserts a request signal (REQ)

when it wants to take control of the bus. The arbiter sends a grant signal (GNT) to the master when permission is granted to take control of the bus. Referring to Figure 2, in a PCI bus, each master in request/grant architecture 200 has its own lines to request bus control and receive confirmation that control has been granted. Thus, each master has its own dedicated REQ# and GNT# lines, wherein # designates the master. When several masters request PCI bus ownership, each requesting master 101/103 asserts its respective request line. For example, master 101B makes its request using dedicated request line REQ#2. Arbiter 103 determines which master should get ownership of PCI bus 106 (Figure 1) and asserts the grant line associated with that master. For example, master 101B receives its grant signal from arbiter 103 via line GNT#2 (note that, for convenience, both the request/grant lines and the signals thereon are referred using the same designators).

In accordance with the present invention, dynamic information, such as the state of the target the master is accessing and the availability of the data the master device is requesting, is incorporated into the arbitration algorithm, thereby greatly enhancing system performance and maximizing system efficiency. Specifically, the arbiter logic in the present invention includes multiple levels of master priority.

In one embodiment, three levels of priority are provided: HIGH, MEDIUM, and LOW. As explained in further detail below, a master that has a request posted in the request queue of the target, but does not have data available in the target, has a LOW priority. Any master which does not have its request posted in the request queue of the target has a MEDIUM priority. Finally, a master

1 that has a request posted in the request queue of the
2 target and data is available in the target has a HIGH
3 priority.

4 Figure 3 illustrates a flow chart 300 that summarizes
5 the priority scheme for each master in accordance with the
6 present invention. After the prioritization process begins
7 in step 301, the arbiter assigns a MEDIUM priority to the
8 master in step 302. At this point, the master is inactive
9 until the master asserts a request signal when it wants to
10 take control of the bus in step 303. After the arbiter
11 sends a provisional grant signal to the master in step 304,
12 the arbiter determines whether data is available from the
13 target associated with the request in step 305.

14 Assuming data is not available, then the arbiter
15 assigns a LOW priority to the master in step 306. The
16 master maintains this priority until the target data is
17 available, as determined in step 307. At this point, the
18 request posted by the master is modified in step 308 (i.e.,
19 a master ID is generated) and sent to the arbiter. After
20 receiving the modified request, the arbiter changes the
21 priority of the master to a HIGH priority in step 309.

22 Assuming the PCI bus is available and no other HIGH
23 priority masters have earlier rights to the bus as
24 determined in step 310, the arbiter sends a final grant
25 signal to the master, thereby allowing the master to take
26 control of the PCI bus. After data is transferred from the
27 target in step 311, the arbiter returns to step 302 and
28 changes the priority of the master back to a MEDIUM
29 priority. Note that if target data is available in step
30 305, then the arbiter immediately modifies the request in
31 step 308 and then proceeds through steps 309-311 as
32 described above.

1 Figure 4 illustrates an example prioritization process
2 in which masters 101A, 101B, 101C, and 105 (the
3 microprocessor) each request bus control. At time t1,
4 master 105 requests bus control. Therefore, the arbiter
5 assigns master 105 a MEDIUM priority. At time t2, master
6 101A requests bus control at the same time that data
7 becomes available to master 105. In accordance with the
8 present invention, the arbiter assigns master 101A a MEDIUM
9 priority and changes the priority of master 105 to a HIGH
10 priority. Thus, master 105 is given control of the bus at
11 time t2 until completion of its process at time t5.

12 In the interim, data is not available to master 101A
13 at time t3. Therefore, at time t3, the arbiter reassigns
14 master 101A a LOW priority. Also at time t3, master 101B
15 requests bus control and is assigned a MEDIUM priority. At
16 time t4, data is available to master 101B. Therefore, the
17 arbiter reassigns master 101B a HIGH priority and allows
18 master 101B to take control of the bus at time t5 (i.e.
19 after the process associated with master 105 is complete).
20 The process performed by master 101B ends at time t6.

21 Note that at time t5, master 101C requests bus
22 control. However, because data is not available to master
23 101C at time t6, the arbiter reassigns master 101C a LOW
24 priority. Thus, at this time, both masters 101A and 101C
25 have a LOW priority. At time t7, data is available to
26 master 101C. Therefore, the arbiter reassigns master 101C
27 a HIGH priority and allows master 101C to take control of
28 the bus at time t7.

29 At time t7, master 105 requests control of the bus and
30 is therefore assigned a MEDIUM priority. At time t8, data
31 becomes available to both masters 101A and 105. In
32 accordance with the present invention, master 105

1 (previously having a MEDIUM priority) will take control of
2 the bus before master 101A (previously having a LOW
3 priority). Specifically, master 105 will take control at
4 time t9 when the process performed by master 101C is
5 completed. Master 101A will take control of the bus after
6 the process performed by master 105 is complete.

7 In accordance with the present invention, each target
8 102 includes a memory interface to facilitate the
9 prioritization process. **For example,** referring to Figure
10 2, target 102A includes an interface 201A to its associated
11 memory 104A and target 102B includes an interface 201B to
12 its associated memory 104B. The following example
13 describes the functionality of memory interface 201A,
14 wherein the functionality of memory interface 201B is
15 identical to that of memory interface 201A. Suppose master
16 101A requests access to memory 104A. A request REQ#1 is
17 sent to arbiter 103, which assigns master 101A a MEDIUM
18 priority and forwards the request to target 102A. In one
19 embodiment, request REQ#1 is sent to target 102A via memory
20 bus 107a (Figure 1). However, in other embodiments,
21 request REQ#1 is sent to target 102A via a dedicated line,
22 just as the request/grant signals. Upon receipt of REQ#1,
23 target 102A proceeds to capture the information needed to
24 complete the access of memory 104A.

25 The captured information is stored in a request queue
26 in the interface 201A of target 102A. Figure 5 illustrates
27 an exemplary request queue 500 including request
28 information 501, a plurality of master identifications
29 (IDs) 502, and data 503. In one embodiment, target 102A
30 generates a master ID 502A using request REQ#1 (i.e.,
31 request information 501A) after data 503A is available and
32 stored in request queue 500. Master ID 502A, shown as

1 modified request MREQ#1 in Figure 2, is then provided to
2 arbiter 103. In one embodiment, modified request MREQ#1 is
3 sent to arbiter 103 via memory bus 107a (Figure 1).
4 However, in other embodiments, modified request MREQ#1 is
5 sent to arbiter 103 via a dedicated line, just as the
6 request/grant signals.

7 After arbiter 103 receives modified request MREQ#1,
8 arbiter 103 changes the priority of master 101A to HIGH
9 and, and assuming PCI bus 106 (Figure 1) is available and
10 no other masters have an earlier HIGH priority, sends a
11 final grant signal GNT#1 to master 101A, thereby allowing
12 master 101A to take control of PCI bus 106. Note that if
13 PCI bus 106 is currently being controlled by another master
14 or if other masters have an earlier HIGH priority, then
15 arbiter 103 allows master 101A to take control of PCI bus
16 106 at the next earliest time period after the process
17 performed by the last controlling master is complete.

18 As described in reference to Figure 4, if data is not
19 available in the next time period (time t3) following
20 request REQ#1 (time t2), then arbiter 103 changes the
21 priority of master 101A to LOW. In this manner, master
22 101A need not repeat, and indeed in a preferred embodiment
23 is prevented from repeating, REQ#1 if target 102A is not
24 ready to supply the data. As described previously, during
25 the period that master 101A is assigned a LOW priority (in
26 Figure 4, from time t3 to time t8), arbiter 103 allocates
27 the ownership of PCI bus 106 to another PCI master
28 requesting the bus and having data available (such as
29 master 101B at time t5 and master 101C at time t7).

30 In accordance with one embodiment of the present
31 invention, target 102A can post different requests into the
32 request queue of interface 201A. In this manner, multiple

1 masters can access the memory of the same target (In fact,
2 because of the sequential servicing of the present
3 invention, multiple masters can even access the same data
4 in a memory.) Elaborating upon the example including
5 master 101A above, assume that master 101B also sends a
6 request REQ#2 to arbiter 103 to access memory 104A of
7 target 102A. In this case, target 102A posts REQ#2 (i.e.,
8 request information 501B) from master 101B in request queue
9 500 (Figure 4). Note that although only two requests can
10 be processed by request queue 500, other embodiments can
11 include any number of requests with corresponding master
12 Ids and data. Assume further that the data requested by
13 master 101B is available before the data requested by
14 master 101A. In the present invention, a modified request
15 signal MREQ#2 (i.e., Master ID 502B of Figure 5) (neither
16 shown in Figure 2) is provided to arbiter 103 before
17 modified request MREQ#1 is provided. Thus, the priority of
18 master 101B changes to HIGH before that of master 101A.

19 To further increase the efficiency of the present
20 invention, the request queue may include an input/output
21 cache. The cache interposes a block of fast memory between
22 a memory 104 and arbiter 103. A cache controller keeps the
23 cache filled with data or instructions that one or more
24 masters are most likely to need next. In this manner,
25 information can be retrieved without delay (see Figure 4,
26 for example, in which master 105 posts a request at time t1
27 and retrieves information at time t2). For most system
28 applications, a cache size of approximately 256 KB is
29 satisfactory.

30 In one embodiment, a PCI bridge implements the above-
31 described priority scheme of the present invention. As
32 known by those skilled in the art, a bridge is a system

1 building block used to transport data between various
2 buses. A bridge can connect different types of buses.
3 Thus, referring to Figure 1, a bridge can interconnect PCI
4 bus 106, memory buses 107, and local bus 108. A bridge may
5 be an ASIC device, or may be part of a chipset in the
6 system. In a preferred embodiment, the PCI bridge includes
7 a chipset that integrates targets 102 and arbiter 103.

8 In summary, the present invention includes a priority
9 scheme implemented by the arbiter in conjunction with the
10 memory interfaces in the target devices. The priority of
11 various requesting masters changes dynamically based on
12 data availability and request state. Using the above
13 features, the present invention maximizes utilization of
14 the PCI bandwidth and throughput.

15 The specific embodiments of the present invention are
16 presented for purposes of description and illustration
17 only. These embodiments are not intended to be exhaustive
18 or to limit the invention in any way. Those skilled in the
19 art will recognize modifications and variations to the
20 present invention. The present invention is only defined
21 by the appended claims.

1 CLAIMS

2 I claim:

3 1. A method of assigning ownership of a peripheral
4 component interconnect (PCI) bus, the method including:

5 identifying a target requested by a master;
6 determining if data associated with the target is
7 available; and

8 assigning a priority to the master based on
9 availability of the data.

10
11 2. The method of Claim 1, wherein after the master
12 requests a target, the master is assigned a MEDIUM
13 priority.

14
15 3. The method of Claim 1, wherein if the data is not
16 available, then the master is assigned a LOW priority.

17
18 4. The method of Claim 1, wherein if the data is
19 available, then the master is assigned a HIGH priority.

20
21 5. The method of Claim 1, wherein the target uses
22 delayed transactions to complete a read access.

23
24 6. The method of Claim 5, wherein the target
25 integrates a buffer management scheme.

26
27 7. The method of Claim 6, wherein the buffer
28 management scheme includes an input/output cache.

29
30 8. The method of Claim 1, wherein identifying a
31 target includes sending a request signal from the master to
32 an arbiter.

1

2 9. The method of Claim 8, wherein assigning a
3 priority includes sending a modified request signal to the
4 arbiter.

1 PERIPHERAL COMPONENT INTERCONNECT ARBITER IMPLEMENTATION
2 WITH DYNAMIC PRIORITY SCHEME

3 Sujith Arramreddy

4 Ragu Appanagari
5

6 ABSTRACT OF THE INVENTION

7 A dynamic priority scheme is provided that uses
8 information including the status of the target and data
9 availability in deciding which PCI master should be
10 assigned ownership of the bus. The target uses delayed
11 transactions to complete a read access targeted to it. The
12 target also integrates a buffer management scheme, in one
13 embodiment an input/output cache, for buffer management.
14 The present invention optimizes the performance and
15 utilization of the PCI bus.

Figure 1

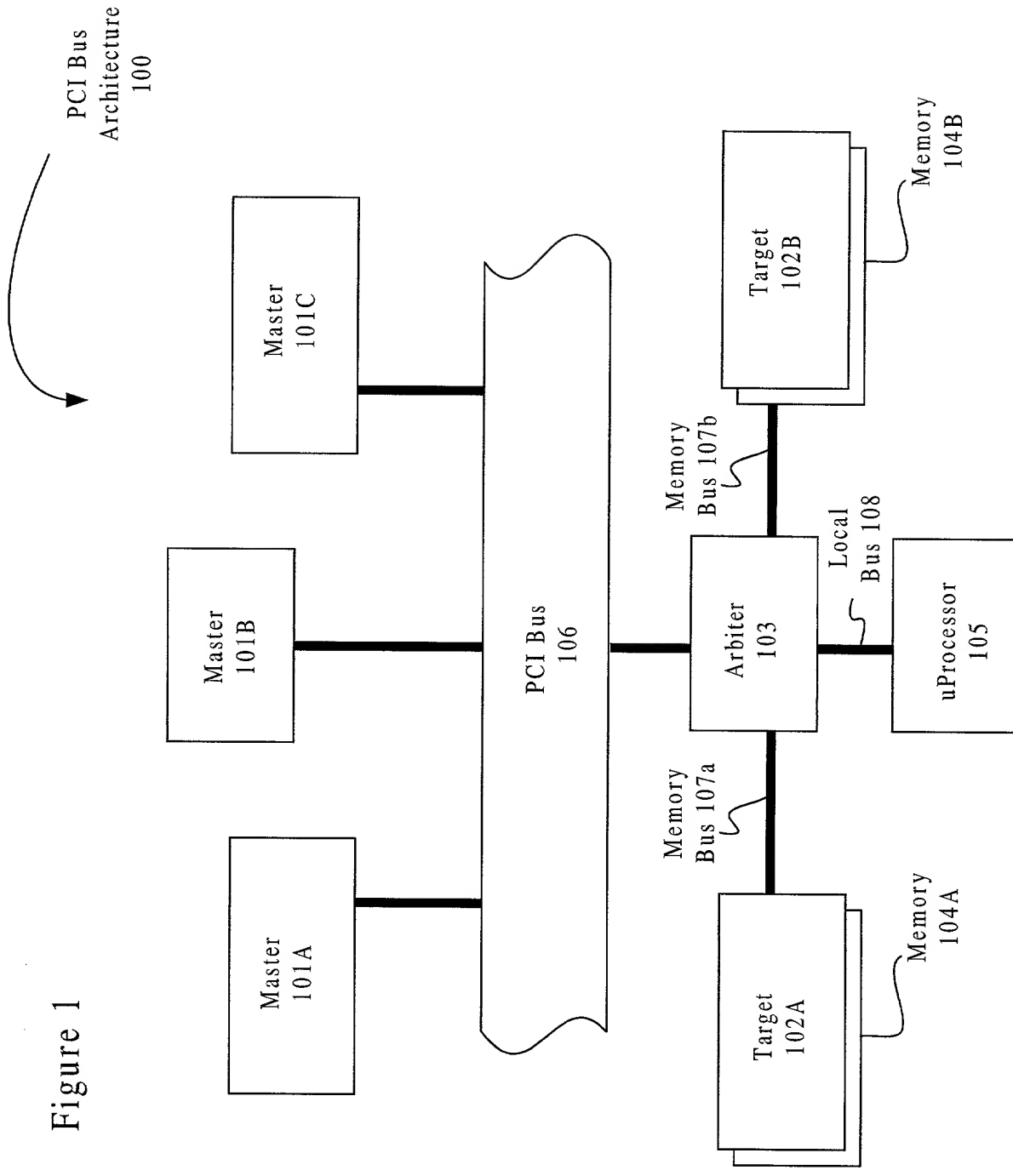
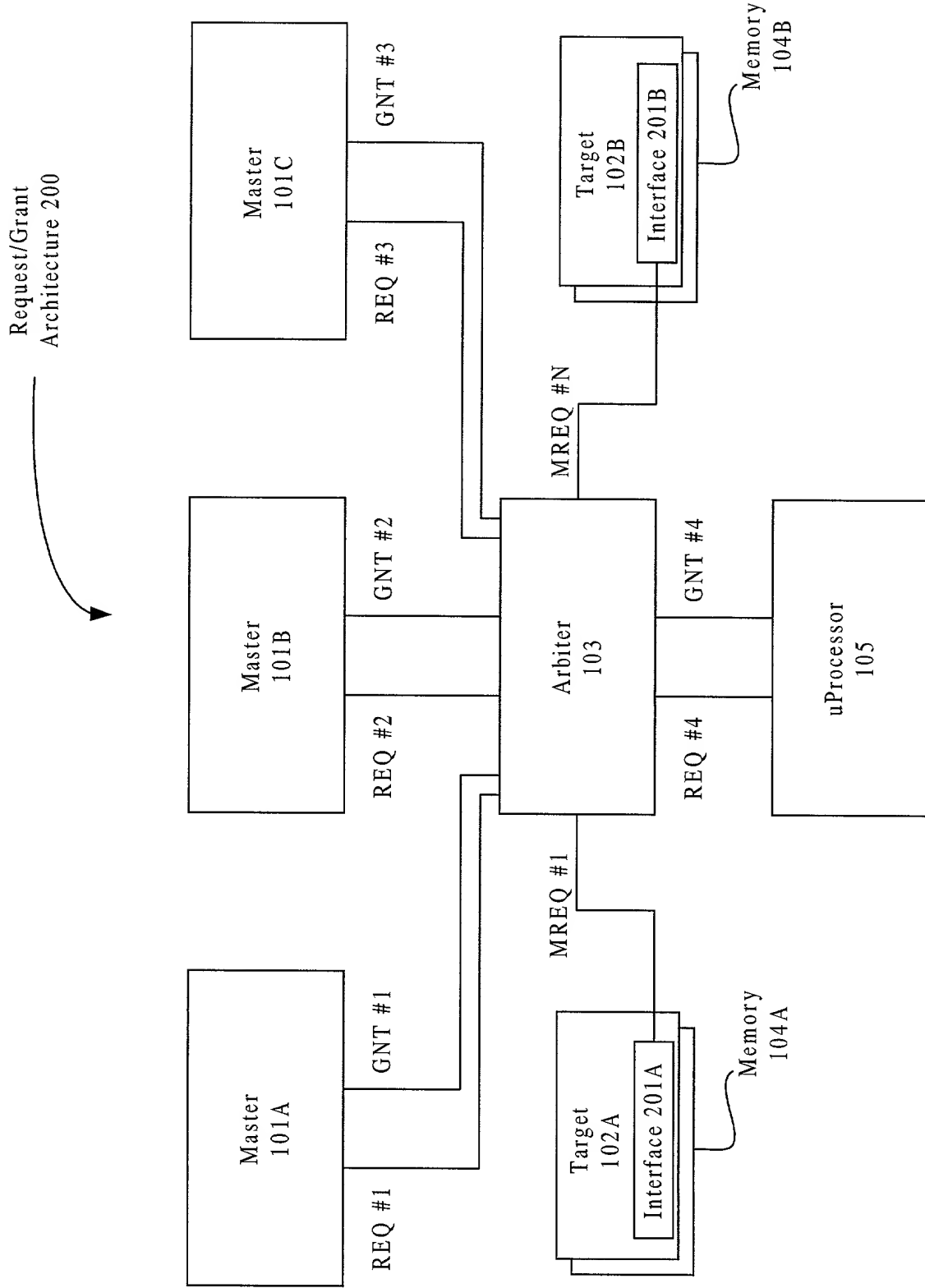


Figure 2



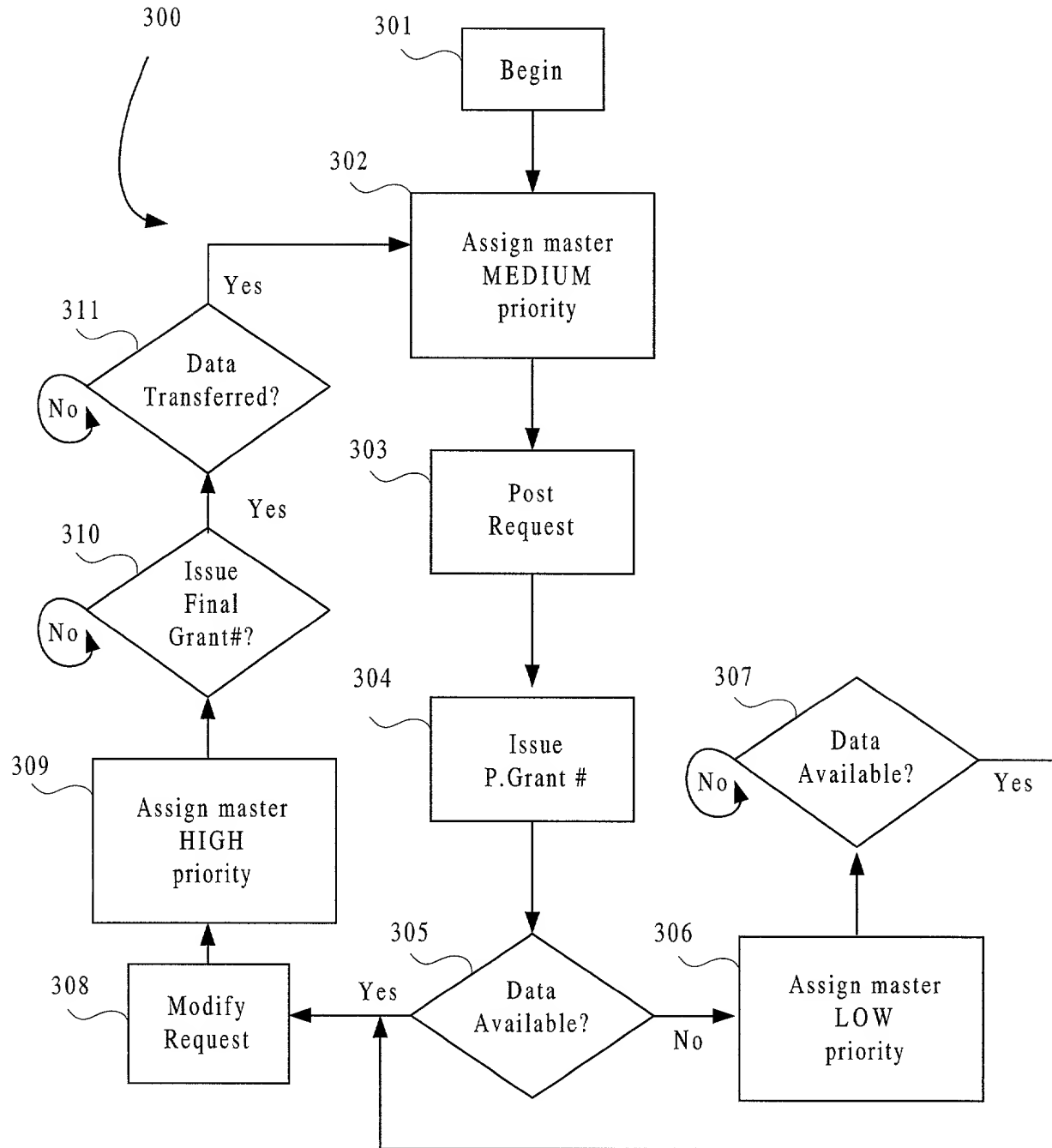


Figure 3

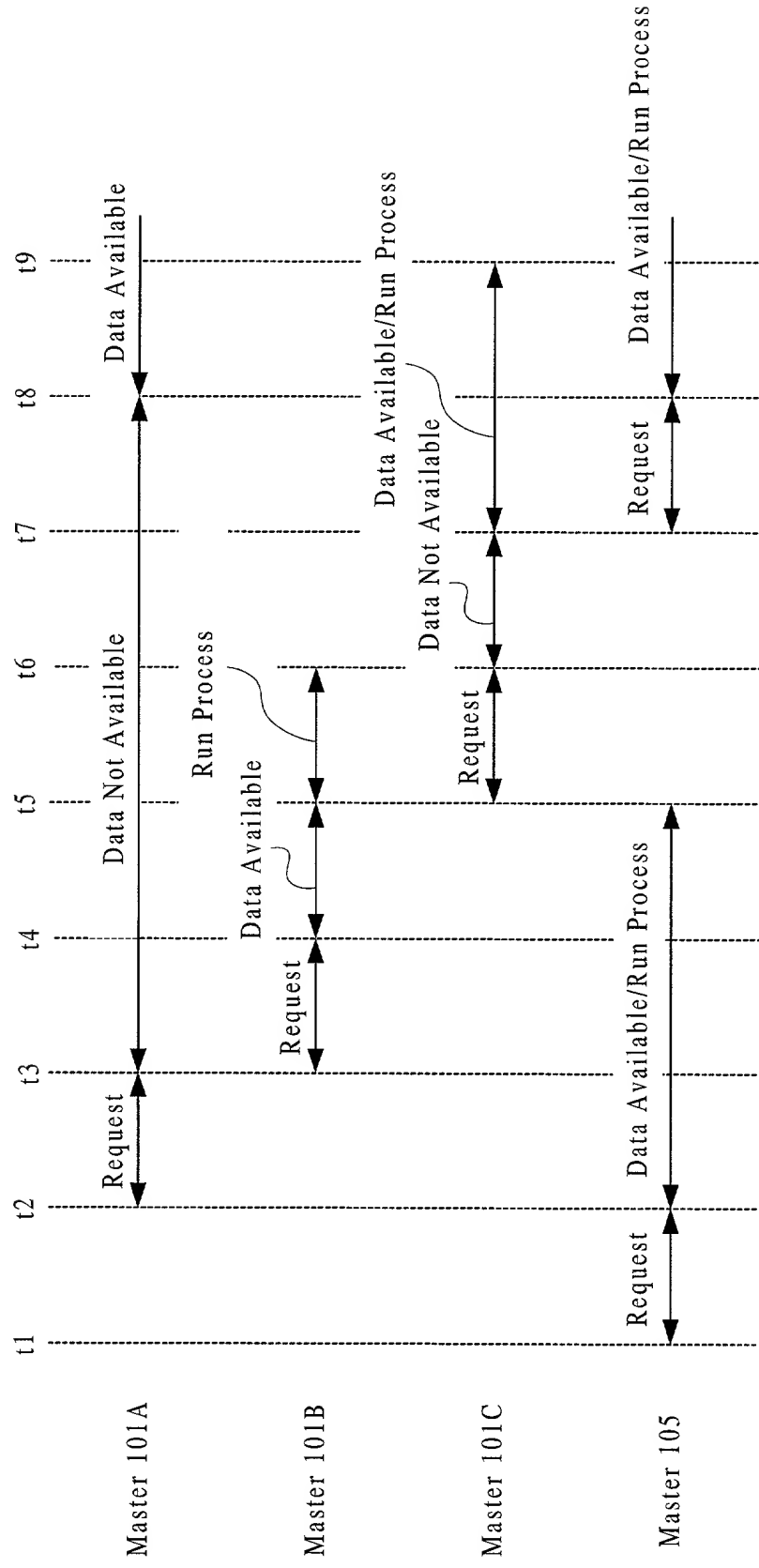


Figure 4

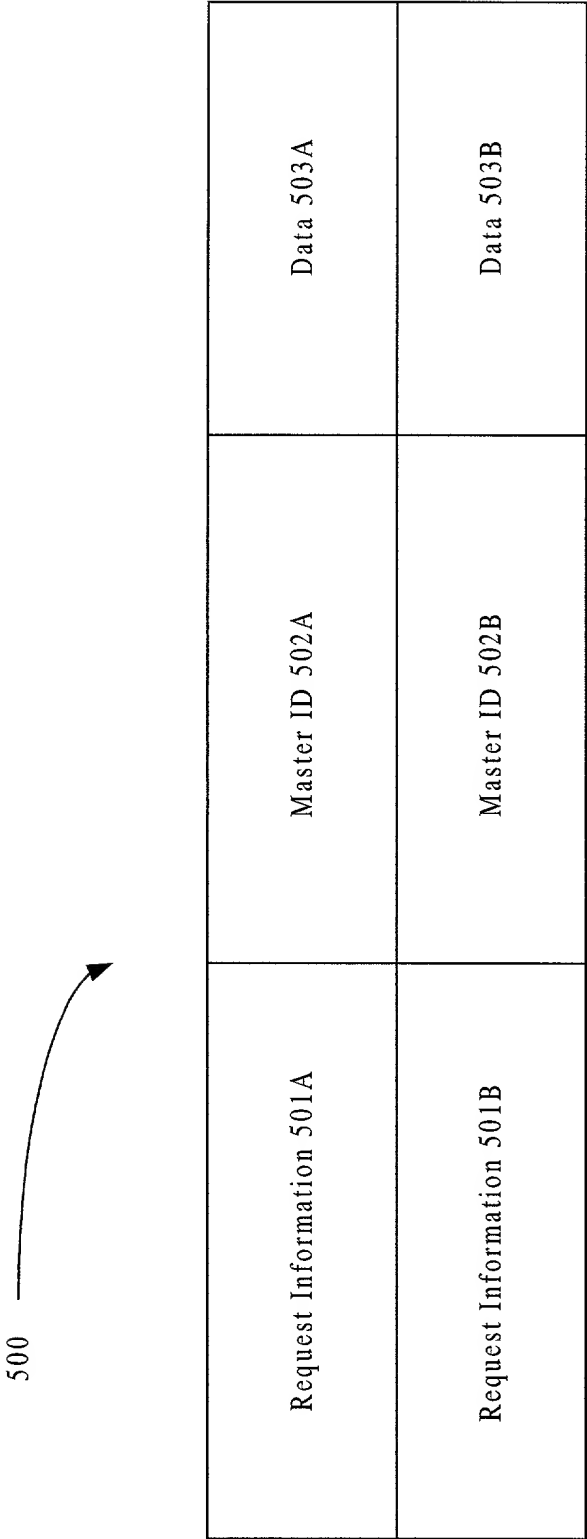


Figure 5